1. **(Currently amended)** A portable thread environment comprising:

an application programming interface configured to support multiple application program tasks, wherein each task is either a preemptive task comprised of preemptive threads or a cooperative task comprised of cooperative threads;

host adaptation logic for communicatively interfacing said cooperative tasks, preemptive tasks, cooperative threads and preemptive threads with a host processing environment; and

a scheduler ~~**configured to**~~ **operable to:**

~~**determine an execution order of cooperative threads and preemptive threads based on each cooperative thread's and preemptive thread's priority levels;**~~

receive a request for a requested thread assigned to a first task;

suspend the currently running thread if the requested thread is not a cooperative thread or if the currently running thread is not assigned to the first task; and

continue running the currently running thread if the requested thread is a cooperative thread and the currently running thread is assigned to the first task.

2. **(Currently amended)** The portable thread environment as in claim 1 wherein threads of the same preemptive task or cooperative task **may have** ~~**have**~~ different priority levels.

3. **(Currently amended)** The portable thread environment as in claim 1 wherein **the scheduler is further operable to determine whether the requested thread has a higher priority level than the currently running thread and wherein the scheduler is operable to suspend the currently running thread by suspending the currently running thread if:**

a) the requested thread has a higher priority level than the currently running thread; and

b) that the requested thread is a preemptive thread or that the currently running thread is not assigned to the first task ~~**a currently running preemptive thread is suspended by a higher priority thread**~~.

4. **(Canceled)**

5. **(Canceled)**

6. **(Canceled)**

7. **(Canceled)**

8. **(Currently amended)** The portable thread environment as in claim 1 wherein **the scheduler is further operable to change a priority level of** a first thread ~~**may change a priority level of**~~ **based on a request generated while running** a second thread.

9. **(Currently amended)** A method for porting an application from a first host environment to a second host environment, **said second host environment having application-specific hardware that supports a first set of functions, and said method** comprising:

modeling**, with a first set of tasks within said application, said** functions **that are** supported by **said** application-specific hardware ~~in the form of a first set of tasks within said application~~**, wherein each task comprises one or more program fragments**;

removing said first set of tasks from said application;

loading said application without said first set tasks to said second host environment, said application-specific hardware providing said functions provided by said first set of tasks in said first host environment; and

configuring said first set of tasks and second set of tasks to communicate by passing a set of messages in said first host environment, wherein one or more of said set of messages are also used to provide communications between said first set of tasks and said application-specific hardware in said second host environment.

10. **(Original)** The method as in claim 9 further comprising:

interrupting the first set of tasks.

11. **(Original)** The method as in claim 10, wherein interrupting the first set of tasks further comprises:

determining an execution order of cooperative threads and preemptive threads based on each cooperative thread's and preemptive thread's priority levels.

12. **(Original)** The method as in claim 11, further comprising:

sustaining a currently running preemptive thread when a cooperative thread or preemptive thread of lower or equal priority is requested.

13. **(Original)** The method as in claim 11, further comprising:

suspending a currently running cooperative thread when a preemptive thread is requested.

14. **(Original)** The method as in claim 11, further comprising:

sustaining a currently running cooperative thread of a task when a requested cooperative thread of the task is requested.

15. **(Original)** The method as in claim 11, further comprising:

suspending a currently running cooperative thread of a first task when a requested cooperative thread of a second task and having a higher priority level is requested.

16. **(Currently amended)** A portable application environment, comprising:

means for modeling**, with a first set of tasks within an application, a first set of** functions **that are** supported by ~~**said**~~ application-specific hardware **in a second host environment** ~~**in the form of a first set of tasks within said application**~~;

means for removing said first set of tasks from said application;

means for loading said application without said first set task to said second host environment, said application-specific hardware providing said functions provided by said first set of tasks in **a first** ~~**said first**~~ host environment;

means for configuring said first set of tasks and second set of tasks to communicate by passing a set of messages in said first host environment, wherein one or more of said set of messages are also used to provide communication between said first set of tasks and said application-specific hardware in said second host environment; and

means for passing the set of messages from a first thread to a second thread.

17. **(Original)** The portable application environment as in claim 17 further comprising:

means for interrupting the first set of tasks.

18. **(Original)** The portable application environment as in claim 17 further comprising:

means for determining an execution order of cooperative threads and preemptive threads based on each cooperative thread's and preemptive thread's priority levels.

19. **(Original)** The portable application environment as in claim 17 further comprising:

means for sustaining a currently running preemptive thread when a cooperative thread or preemptive thread having a lower or equal priority is requested.

20. **(Original)** The portable application environment as in claim 17 further comprising:

means for suspending a currently running preemptive thread when a higher priority cooperative thread or higher priority preemptive thread is requested.

21. **(Original)** The portable application environment as in claim 17 further comprising:

means for suspending a currently running cooperative thread when a higher priority preemptive thread is requested.

22. **(Original)** The portable application environment as in claim 17 further comprising:

means for sustaining a currently running cooperative thread of a task when a requested cooperative thread of the task is requested.

23. **(Original)** The portable application environment as in claim 17 further comprising:

means for suspending a currently running cooperative thread of a first task when a requested cooperative thread of a second task having a higher priority level is requested.

24. **(Currently amended)** A computer-readable medium having stored thereon a plurality of instructions, said plurality of instructions when executed by a computer, cause said computer to perform:

modeling**, with a first set of tasks within an application, a first set of** functions **that are** supported by ~~said~~ application-specific hardware **in a second host environment** ~~in the form of a first set of tasks within said application~~;

removing said first set of tasks from said application;

loading said application without said first set tasks to said second host environment, said application-specific hardware providing said functions provided by said first set of tasks in said first host environment;

configuring said first set of tasks and second set of tasks to communicate by passing a set of messages in said first host environment, wherein one or more of said set of messages are also used to provide communication between said first set of tasks and said application-specific hardware in said second host environment; and

passing the set of messages from a first thread to a second thread.

25. **(Original)** The computer-readable medium of claim 24 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:

interrupting the first set of tasks.

26. **(Original)** The computer-readable medium of claim 25 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:

determining an execution order of cooperative threads and preemptive threads based on each cooperative thread's and preemptive thread's priority levels.

27. **(Original)** The computer-readable medium of claim 25 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:

sustaining a currently running preemptive thread when a cooperative thread or preemptive thread having lower or equal priority is requested.

28. **(Original)** The computer-readable medium of claim 25 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:

suspending a currently running preemptive thread when a higher priority cooperative thread or higher priority preemptive thread is requested.

29. **(Original)** The computer-readable medium of claim 25 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:

suspending a currently running cooperative thread when a higher priority preemptive thread is requested.

30. **(Original)** The computer-readable medium of claim 25 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:

sustaining a currently running cooperative thread of a task when a requested cooperative thread of the task is requested.

31. **(Original)** The computer-readable medium of claim 25 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:

suspending a currently running cooperative thread of a first task when a requested cooperative thread of a second task and having a higher priority level is requested.